

Bounding XCS's Parameters for Unbalanced Datasets

Albert Orriols-Puig
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull
Quatre Camins, 2. 08022 Barcelona, Spain.
aorriols@salleurl.edu

Ester Bernadó-Mansilla
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull
Quatre Camins, 2. 08022 Barcelona, Spain.
esterb@salleurl.edu

ABSTRACT

This paper analyzes the behavior of the XCS classifier system on imbalanced datasets. We show that XCS with standard parameter settings is quite robust to considerable class imbalances. For high class imbalances, XCS suffers from biases toward the majority class. We analyze XCS's behavior under such extreme imbalances and prove that appropriate parameter tuning improves significantly XCS's performance. Specifically, we counterbalance the imbalance ratio by equalizing the reproduction probabilities of the most occurring and least occurring niches. The study provides guidelines to tune XCS's parameters for unbalanced datasets, based on the dataset imbalance ratio. We propose a method to estimate the imbalance ratio during XCS's training and adapt XCS's parameters online.

Categories and Subject Descriptors

I.2.6 [Learning]: concept learning, knowledge acquisition

General Terms

Experimentation

Keywords

Evolutionary Computation, Genetic Algorithms, Machine Learning, Learning Classifier Systems, Class Imbalance

1. INTRODUCTION

During the last decades, research on evolutionary computation (EC) and learning classifier systems (LCSs) has led to better knowledge and improved applicability of LCSs. Recently, LCSs and specially XCS [10, 11] have been enhanced to solve challenging machine learning problems. Learning from imbalance datasets has been identified as one of such current challenging problems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
Copyright 2006 ACM 1-59593-186-4/06/0007 ...\$5.00.

A dataset is said to be imbalanced if one of the classes is represented by a very small number of examples compared to the other classes. This case occurs in several domains such as fraud detection, oil spills in satellite images, failures in manufacturing process, rare medical diagnosis, etc. Many learners assume a uniform distribution of classes, so that they may suffer from biases toward the majority class when they are exposed to high class imbalances. Japkowicz [7] demonstrated empirically that C5.0, and SVM to a minor degree, suffered from class imbalances.

Methods to deal with class imbalances can be applied at the sampling level or at the classifier level. Those acting at the sampling level aim at balancing the a priori probabilities of classes, either by oversampling the minority class instances or undersampling the majority class instances. The second class of methods tries to adapt the classifier to class imbalances, e.g., by measuring each classification cost separately.

In the LCS field, there are few studies analyzing LCS's behavior with imbalanced datasets. Holmes [5] addressed this topic in the context of epidemiological datasets, and adapted EpiCS applying an strategy based on disproportionate reinforcement per class. Other recent studies [8, 9] analyzed UCS [1], a supervised LCS derived from XCS, with different levels of class imbalances. Results showed that the system favored the majority class with conditions of severe class imbalances. Several mechanisms both at the sampling level and at the classifier level were proved useful to minimize the bias.

In the field of XCS, there are no systematic studies on the influence of unbalanced datasets to XCS's performance. Although UCS and XCS share several features, parameter updates and fitness computation are different, so that the behavior and dynamics of UCS may not be directly extended to XCS. Some preliminary results [12] seem promising and indicate that XCS has little difficulty learning problems with considerable class imbalance. Our aim is to extend these results and investigate XCS's behavior with unbalanced datasets.

To understand XCS's response to different levels of class imbalance, we use the multiplexer problem, which we proceed to unbalance gradually. We derive theoretical dataset bounds necessary for XCS to learn both classes. We find that XCS is quite robust to moderate class imbalances, but it performs poorly for high class imbalances even when the dataset bounds are satisfied. Then, we identify relevant XCS's parameters and provide guidelines to set them appropriately under such extreme conditions. In particular, we

can counterbalance the imbalance ratio between the majority and the minority class instances by adjusting the genetic opportunities of their respective niches. The study reveals that an appropriate setting of XCS’s parameters notably improves XCS’s performance for high class imbalances. However, we acknowledge that some parameters might be difficult to estimate for real-world problems with unknown imbalance ratios. Therefore, we propose a method that adapts these parameters automatically, based on information collected during XCS’s search. The study optimizes XCS for unbalanced datasets and sets the framework for the application of other sampling or classifier methodologies, which could even improve performance and possibly speed up convergence.

The paper is organized as follows. Section 2 shows the results of XCS with the unbalanced multiplexer. Section 3 analyzes XCS’s behavior and revises parameter bounds for XCS with imbalanced datasets. Then, we rerun XCS with appropriate settings. Next, we propose a method to adjust these values automatically and show its results. Section 5 discusses extensions of the current work and finally, we provide conclusions and future work.

2. XCS ON UNBALANCED MULTIPLEXER

Our first concern was to analyze the effects of training XCS with datasets that contained class imbalances. For this purpose, we ran XCS with the 11-bit multiplexer problem for different imbalance levels. The unbalancing process was made as in [12]. At each learning step, an input example was randomly chosen. If it belonged to the class labeled as the *minority class*, it was accepted with sampling probability sp . If it was discarded, another input example was chosen, which underwent the same process. In all experiments made herein, the undersampled class was the one labeled as “1”, whereas the class labeled as “0” was sampled with the normal rate.

We gradually unbalanced the multiplexer problem to different degrees. We will denote as i the *imbalance level* of the dataset, where $sp = 1/2^i$. Thus, level $i = 0$ represents the balanced multiplexer. For level $i > 1$, there are half of the minority class instances with respect to $i - 1$. In the following, the *imbalance ratio* ir will refer to the proportion of the majority class instances with respect to the minority class instances.

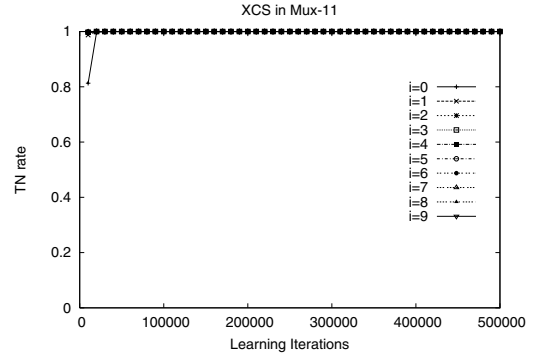
XCS¹ was implemented as described in [4]. The parameters used in all runs were:

$$N=800, \beta=0.2, \alpha=0.1, \epsilon_0 = 1, \nu=5, \theta_{GA}=25, \\ \chi=0.8, \mu=0.04, \theta_{del}=20, \delta=0.1, \theta_{sub}=200, P_{\#}=0.6$$

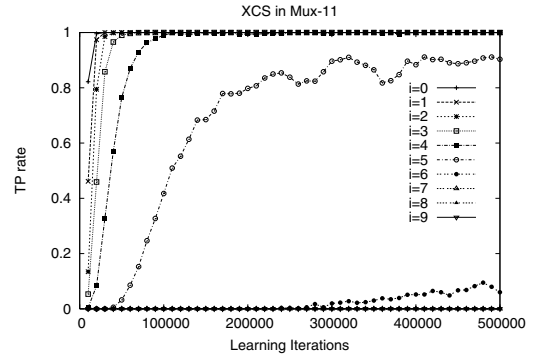
We used proportionate selection and niched mutation (as introduced in [4]). To avoid a high generalization pressure, subsumption was not applied on the action set, and was applied on the GA but requiring a subsumer classifier to be highly experienced by setting $\theta_{sub} = 200$.

Figure 1 shows the true negative (TN) rate, i.e., the ratio of correct classifications for class 0, and the true positive (TP) rate, the ratio of correct classifications for class 1, obtained by XCS with imbalance levels from $i=0$ to $i=9$. Curves are averaged over ten runs. In all runs, XCS was trained during 5,000,000 learning iterations, but only the

¹We assume the reader has familiarity with the XCS classifier system. Otherwise, the reader is referred to [10, 11, 4].



(a) TN rate



(b) TP rate

Figure 1: TN rate (a) and TP rate (b) in the 11-bit multiplexer with imbalance levels from 0 to 9.

first 500,000 are shown for a better visibility. Figure 1(a) shows that, for all imbalance levels, the TN rate quickly reaches 100%. Figure 1(b) shows that the TP rate raises to 100% for imbalance levels up to $i=4$. For $i=5$, the TP rate is 90% after $5 \cdot 10^5$ iterations, stabilizing at 100% after $2 \cdot 10^6$ explore trials. However, for $i=6$, the TP rate remains below 20%, and increasing the number of learning iterations up to $5 \cdot 10^6$ provides no improvement. For higher imbalance levels, from $i=7$ to $i=9$, the system classifies all the input instances as if they belonged to the majority class. Table 1 shows the most numerous rules evolved by XCS for $i=7$. The population mainly consists of the two most overgeneral rules.

The results show that the TP rate converges more slowly as the imbalance level increases. At a given point, XCS is not able to classify correctly the minority class instances and evolves overgeneral classifiers. The next section analyzes the possible reasons by reviewing XCS’s behavior and the influence of parameter settings in XCS’s performance under unbalanced datasets.

3. ANALYSIS OF PARAMETER BOUNDS

3.1 Classifiers’ Error in Unbalanced Datasets

Our first concern is whether overgeneral classifiers can oc-

Table 1: Most numerous rules evolved in a run of XCS with the 11-bit multiplexer for $i=7$. Cond. in the classifier’s condition, A. the action it predicts, and P., Err., F. and Num. are the prediction, error, fitness and numerosity of the classifier.

Cond.	A.	P.	Err.	F	Num.
#####	0	1000	$1.2 \cdot 10^{-4}$	0.98	385
#####	1	$1.2 \cdot 10^{-4}$	$7.4 \cdot 10^{-5}$	0.98	366
...					

cur easily with unbalanced datasets. If the ratio of instances of the majority class with respect to the minority class increases, overgeneral classifiers that match instances of both classes and predict the majority class will tend to have lower error. At a given imbalance ratio, overgeneral classifiers will be considered as accurate. This will happen when their error is less than ϵ_0 . We seek to determine the imbalance ratio bound that allow overgeneral classifiers to be identified as inaccurate.

According to [2], the prediction P of a classifier can be approximated by:

$$P = P_c(cl) \cdot R_{max} + (1 - P_c(cl)) \cdot R_{min} \quad (1)$$

where $P_c(cl)$ is the probability that a classifier classifies the matching input correctly, R_{max} is the maximum reward, and R_{min} the minimum reward given by the environment. The error of a classifier can be approximated by:

$$\epsilon = |P - R_{max}|P_c(cl) + |P - R_{min}|(1 - P_c(cl)) \quad (2)$$

For classification problems, R_{min} is usually 0, so that the prediction of a classifier can be estimated by: $P = P_c(cl)R_{max}$. Substituting P into formula 2, we get a prediction error estimate:

$$\epsilon = 2R_{max}(P_c(cl) - P_c(cl)^2) \quad (3)$$

$P_c(cl)$ can be approximated as:

$$P_c(cl) = \frac{C}{C+!C} \quad (4)$$

where C is the number of instances that the classifier predicts correctly (i.e., the predicted class agrees with that of the sample) and !C is the number of instances that the classifier predicts incorrectly (i.e., the predicted class does not agree with that of the sample). The sum $C+!C$ corresponds to the total number of examples that the classifier matches.

Let’s denote as p the ratio between !C and C:

$$p = \frac{!C}{C} \quad (5)$$

We can derive the classifier’s probability of being correct as:

$$P_c(cl) = \frac{1}{1+p} \quad (6)$$

and its error estimate as:

$$\epsilon = \frac{2p}{(1+p)^2} R_{max} \quad (7)$$

In an accurate classifier, $p = 0$, which gives an error estimate equal to zero. For the maximally overgeneral classifier

predicting the majority class, $p=1/ir$. An overgeneral classifier would be considered inaccurate as long as:

$$\epsilon \geq \epsilon_0 \quad (8)$$

Using equation 7, we get:

$$\frac{2p}{(1+p)^2} R_{max} \geq \epsilon_0 \quad (9)$$

Deriving the formula, we obtain:

$$-\epsilon_0 p^2 + 2p(R_{max} - \epsilon_0) - \epsilon_0 \geq 0 \quad (10)$$

This represents a parabola where ϵ takes values higher than ϵ_0 for p ranging between p_l and p_u .

Formula 10 sets the error bounds for a classifier to be considered as accurate. It depends on both R_{max} and ϵ_0 . Substituting $\epsilon_0 = 1$ and $R_{max} = 1000$ in formula 10, the approximate values for p_l and p_u are: $p_l = 1/1998$, $p_u = 1998$. This means that an overgeneral classifier covering a ratio higher than 1998 instances of one class with respect to the others would be considered as accurate. As long as the ratio of instances of one class does not exceed this bound, an overgeneral classifier would be theoretically detected as inaccurate.

In the unbalanced multiplexer at a given imbalance ratio ir , the overgeneral classifier predicting the majority class #####:0 has a ratio $p = 1/ir$. Thus, its error will be higher than ϵ_0 if:

$$1 \leq ir \leq 1998 \quad (11)$$

which corresponds to an imbalance level $i < 11$. This suggests that XCS would theoretically identify overgeneral classifiers as inaccurate as long as $i < 11$. Nonetheless, XCS found difficulties for $i \geq 6$. For $i = 7$ and higher, the population was overcome by overgeneral classifiers.

Next, we analyze if this could be caused by a deviation of the real values of prediction and error with respect to the theoretical estimates. This deviation could be significant for high values of the learning rate β .

3.2 Learning Rate and Stability of Prediction and Error Estimates

In the 11-bit multiplexer with imbalance level $i = 7$, the population evolved by XCS consisted mostly of two overgeneral classifiers, as shown in table 1. Note that the overgeneral rule #####:0 has prediction $P = 1000$ and a very low error, much lower than ϵ_0 . However, the prediction and error estimates computed for $ir = 128$ are:

$$\epsilon = 15.38 \quad (12)$$

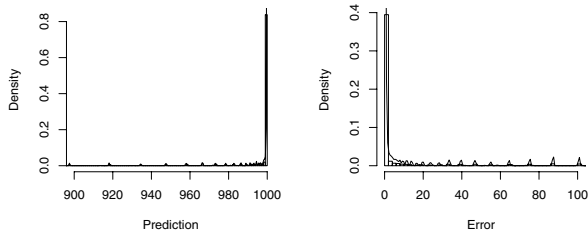
$$P = 992.24 \quad (13)$$

The overgeneral rule #####:1 has a prediction and error very low, but still does not correspond to the theoretical values:

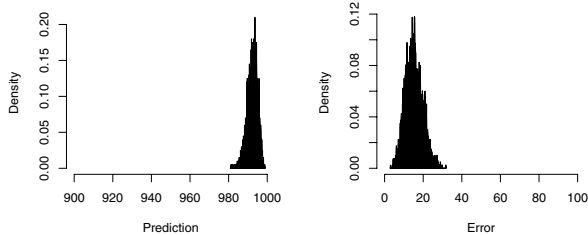
$$\epsilon = 15.38 \quad (14)$$

$$P = 7.75 \quad (15)$$

We attribute the deviation between the real and the theoretical parameters’ values to the chosen β . In the reported experiments $\beta = 0.2$. This is a relatively high value, which may be appropriate to get a fast approximation to the estimates, but tends to oscillate when the classifiers do not receive uniform distribution of samples. In the case of huge



(a) P with $\beta = 0.2$ (b) ϵ with $\beta = 0.2$



(c) P with $\beta = 0.002$ (d) ϵ with $\beta = 0.002$

Figure 2: Evolution of parameters P and ϵ of classifier #####:0 in the 11-bit multiplexer at imbalance level $i=7$: (a) and (b) using $\beta = 0.2$; (c) and (d) using $\beta = 0.002$.

class imbalances, overgeneral classifiers predicting the majority class match one instance of the minority class for each 2^i instances of the majority class. This means that, for 2^i samples, the overgeneral classifier receives the maximum payoff and, in only one case, the overgeneral classifier receives the minimum payoff. For high values of β , such a regime could lead to unstable estimations. In fact, the parameter updates give much more importance to the recent rewards; i.e., the values of the parameters are reflecting the average of few learning iterations (the most recent ones). Decreasing β causes a slower but better approximation of these parameters.

Figure 2 shows the distribution of prediction and error values of the overgeneral classifier #####:0 along a single run, for $\beta = 0.2$ and $\beta = 0.002$. The experiment was performed by initializing the population with the two maximally overgeneral rules—thus, without covering—, and disabling the genetic algorithm. P and ϵ values were sampled every 1,000 learning iterations. The results show that, with $\beta = 0.2$, P takes a value very close to 1000 (see figure 2(a)), and ϵ oscillates near 0 (see figure 2(b)). Besides, there are other peaks in the distribution density, which denote that these values change quickly when the classifier sees a counterexample. Decreasing β to 0.002 smoothes the density curves and the distribution becomes closer to the theoretical values (see figures 2(c) and 2(d)).

Note that the real values of the prediction and error have

a deviation over the theoretical values. This deviation is significant with unequal distribution of examples and counterexamples, plus an additional effect which depends on the setting of the learning rate β . We could rewrite formula 10 to consider the deviation with respect to the theoretical bounds as:

$$\epsilon \pm \sigma > \epsilon_0 \tag{16}$$

To minimize this deviation, we reran the experiments in the 11-multiplexer with $\beta = 0.002$ (not shown for brevity). We found that the classifiers’ parameters were better estimated but the TP rate was not improved with respect to the previous results.

3.3 Occurrence-based reproduction

In this section, we seek to analyze why XCS is not able to create and maintain accurate maximally general rules covering each of the classes for high imbalance ratios. Specifically, we will focus on the reproductive opportunities of niches of the majority and the minority class. As the classifier’s reproduction is occurrence-based, we start by analyzing the classifier’s probability of belonging to an action set, which we denote as p_{occ} (probability of occurrence).

The population evolved with high class imbalances contained two overgeneral classifiers: **s0: #####:0** and **s1: #####:1**. Let’s derive the occurrence probability of these classifiers. During training, XCS runs under an exploration regime: for each explore trial, an action is selected randomly. Therefore, the occurrence probability of these overgeneral classifiers is 1/2. Their participation in an action set does not depend on the imbalance ratio. That is why the numerosity of both overgeneral classifiers is very similar (see table 1), even though the examples of class 1 are less frequent.

Next, let’s analyze two of the accurate maximally general classifiers representing a majority class niche and a minority class niche respectively: $m_0=0000#####:0$ and $m_1=0001#####:1$. The first one classifies the majority class; thus, its occurrence probability is almost the same as in the balanced case. The occurrence probability of the second classifier is inversely proportional to the imbalance ratio ir , since its representative examples belong to the minority class (for brevity, we do not develop the full formulas).

Thus, the occurrence opportunities of overgeneral classifiers with respect to maximally general classifiers (predicting the minority class) is proportional to the imbalance ratio. Similarly, the occurrence opportunities of maximally general classifiers predicting the majority class with respect to those predicting the minority class increase proportionally to the imbalance ratio.

The activation of a classifier triggers its parameters update. Thus, classifiers activating more often have better parameter estimates. Moreover, when a classifier participates in an action set, the genetic algorithm may trigger. Consequently, more frequent classifiers will have more reproduction opportunities. As class imbalance increases, the reproduction opportunities of overgeneral classifiers increase with respect to niches covering the minority class. In the same way, those niches covering the majority class have more reproduction opportunities than their counterparts covering the minority class.

The GA rate depends on the setting of the GA triggering threshold θ_{GA} and the classifier’s occurrence. We recall that

the GA triggers if the average time in the action set since the last GA event is greater than θ_{GA} . Thus, if θ_{GA} is very low, the GA will be activated very often and will favor the most frequent classifiers. Those classifiers occurring infrequently will receive a GA event only when they belong to an action set. This is the case for $\theta_{GA} = 20$, which is the value used in our experiments. A way to counterbalance the reproductive opportunities of niches is to set θ_{GA} to a value higher than the maximum delay between infrequent niches. This would guarantee that all niches will receive the same opportunities, regardless of the occurrence frequency of each niche. Besides, this counteracts the generalization pressure toward overgeneral classifiers.

3.4 Population Size

Population size has been identified as a key parameter for niche support in XCS. The study reported in [3] derives a population size bound that suggests that population size should increase linearly with $1/p_{occ}$. The authors argue that the probability of a niche to be lost depends on the delay between niche occurrences. Setting an appropriate population size guarantees that all niches will be maintained with a given confidence value. In the case of different occurrence probabilities, the study recommends to set p_{occ} to the lowest occurrence probability, but there is no experimental evidence of the potential benefits.

According to this study, population size should be increased with the imbalance of the multiplexer problem. However, all runs made herein were made with the same population size. We reran XCS for $i = 6$ and $i = 7$ with higher population sizes to analyze whether the correct niches could be evolved and maintained in the population. We used the same configuration as in section 2, except for $N=10,000$ at $i=6$, and $N=20,000$ at $i=7$. The results provided no improvement. In fact, XCS's behavior was similar, with overgeneral classifiers arising in the population, with high numerosity and unstable estimates.

This suggests that a compound solution is required for an optimal performance of XCS in unbalanced problems. The next section gives the details.

3.5 Guidelines for Parameter Tuning

XCS's dynamics on unbalanced problems depends highly on parameter settings. Our analysis provides insight into the role of some critical parameters and its influence on XCS's performance. In the following, we summarize some guidelines to set these parameters appropriately.

Firstly, ϵ_0 and R_{max} determine the maximum error that a classifier can have to be considered as accurate. They define the threshold between negligible noise and imbalance acceptance.

The learning rate β determines somehow the window size of the update mechanism. The worst case is that of an overgeneral classifier covering both the most frequent niche and the least frequent niche. A small learning rate sets the window size to include examples of both niches, and thus avoids forgetting the less frequent examples. Otherwise, the parameter estimates would fluctuate and overgeneral classifiers would be considered accurate. Our suggestion is to set β according to the proportion of frequencies between the least occurring niche (f_{min}) and the most frequent niche

(f_{max}):

$$\beta = k_1 \cdot \frac{f_{min}}{f_{max}} \quad (17)$$

where k_1 is an arbitrary constant. The ratio f_{min}/f_{max} tells us how many examples of the minority class niche are provided with respect to those of the majority class niche. If this ratio decreases, we should also decrease β .

Finally, section 3.3 pointed that θ_{GA} should be set to a value greater than the delay between examples of the minority niche:

$$\theta_{GA} = k_2 \cdot \frac{1}{f_{min}} \quad (18)$$

where k_2 is a constant. This counterbalances the opportunities between minority and majority class niches. For $k_2 > 1$, the reproductive events over all niches will be completely balanced, as suggested in section 3.3.

Moreover, the population size should assure that no niche will be lost. As suggested elsewhere [3], it should be increased proportionally with the imbalance ratio ir .

3.6 Results

We reran the same experiments with the 11-bit multiplexer, setting XCS's parameters according to the recommendations. We only changed the parameters of the runs that failed: $i=\{6,7,8,9\}$. Specifically, $\theta_{GA} = \{200, 400, 800, 1600\}$ and $\beta = \{0.04, 0.02, 0.01, 0.005\}$ for each imbalance level respectively. The population size was initially not modified.

The results, reported in figure 3, show that XCS can solve the unbalanced 11-bit multiplexer problem until $i = 8$, which is a notable improvement with respect to the initial experiments. The theoretical dataset bound was $i < 11$ (see section 3.1). We got closer to this bound with the appropriate parameter settings.

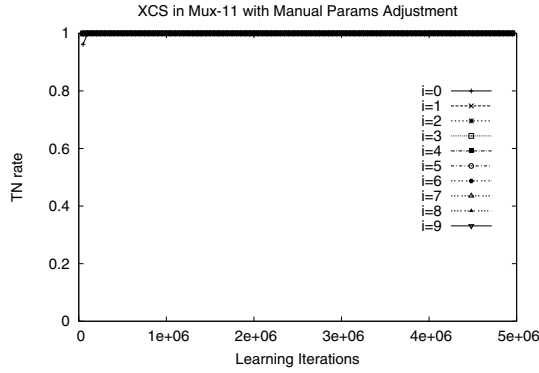
Higher population sizes (not shown for brevity) did not improve these results. In fact, our results prove that setting appropriately the learning rate and the GA triggering threshold suffice for better performance. Higher population sizes are not needed in our case, so that computational resources may be minimized.

4. PARAMETER ADAPTATION

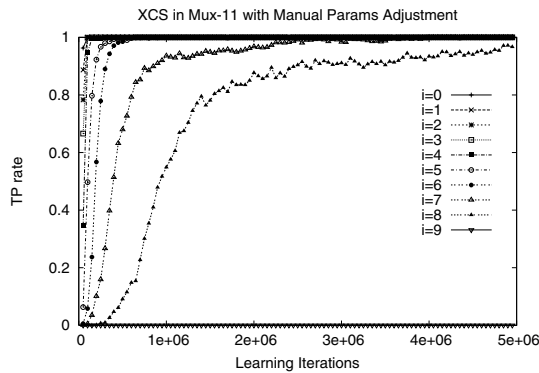
4.1 Reformulating the Problem

The provided guidelines assume that the frequency of minority and majority class niches is known. In the multiplexer problem, these frequencies could be estimated from the imbalance ratio of the dataset. However, in real-world datasets containing class imbalances, niche frequencies are unknown. Although we can estimate the class imbalance ratio from the dataset, this may not be related to niche frequencies. In fact, the class imbalance ratio tells us the proportion of examples per class, but does not provide information about the distribution of niches over the feature space. Even with a balanced ratio of instances per class, XCS (and other learners as well) may suffer from unbalanced niches. This has commonly been addressed as the problem of small disjuncts [6].

Rather than working on class imbalance ratios, we are more focused on niche imbalance ratios. Specifically, we aim at estimating the frequency of the most frequent niche



(a) TN rate



(b) TP rate

Figure 3: TN rate (a) and TP rate (b) in the 11-bit multiplexer with imbalance levels from 0 to 9. Parameters are set according to the guidelines.

(f_{max}) and the least frequent niche (f_{min}). From their estimation, we could derive a bound for the learning rate and the GA triggering threshold.

Our approach is to use information collected by the classifiers themselves during XCS’s training. From the analysis of XCS’s behavior in the multiplexer problem, we assume that overgeneral classifiers will tend to cover niches that are close in the attribute space and present a high imbalance ratio. If we compute the percentage of instances of each class that the overgeneral covers, we can get an estimate of the imbalance ratio between the niches.

This is a local approach to the problem, which we think is much more accurate than the global approach on class imbalance ratio. Thus, we really consider unbalanced niches that promote overgeneral classifiers. The identification of the infrequent niches covered by an overgeneral classifier can be classified as an online identification of the small disjuncts.

4.2 Online Adaptation of Learning Parameters

The procedure identifies the overgeneral classifiers. We estimate the niche imbalance ratio ir_{cl} by computing the ratio between the number of covered instances of the majority

1. ADAPT PARAMETERS
2. if $((p_{cl_{t-1}} > L_{max} \cdot R_{max}) \wedge (p_{cl_t} < L_{dec} \cdot R_{max}))$ then
3. $ir_{cl} := \frac{exp_{maj}(cl)}{exp_{min}(cl)}$
4. if $ir_{cl} > ir_0$ then
5. if $((exp_{cl} > \theta_{ir}) \wedge (num_{cl} > \overline{num}_{[p]}))$ then
6. Adapt β
7. Adapt θ_{GA}
8. end if
9. end if
10. end if

Figure 4: Algorithm firing parameter adaptation. exp_{maj} and exp_{min} are the classifier’s experience in the majority and the minority class respectively. L_{max} and L_{dec} are two constants which determine the degree of oscillation of the prediction parameter firing the adaptation algorithm. ir_0 and θ_{ir} are constants set by the user. The parameters of the current classifier are: exp (experience), num (numerosity) and p (prediction). See text for details.

1. ADAPT β
2. $ir_{cl} := \frac{exp_{maj}(cl)}{exp_{min}(cl)}$
3. do
4. Estimate $p_{ir_{\beta}}$ with the current β
5. if $p_{ir_{\beta}} > p_{th}$ then
6. $\beta = \beta \cdot \zeta$
7. end if
8. while $p_{ir_{\beta}} > p_{th}$

Figure 5: Algorithm for adaptation of parameter β . p_{th} is the value of prediction estimated with formula 6. $p_{ir_{\beta}}$ is an estimation of the value of prediction that the classifier would obtain with the current β . ζ is a discount factor ($\zeta < 1$).

class (exp_{maj}) with respect to those of the minority class (exp_{min}). Note that XCS is slightly modified to compute the experience per class. With that estimate, the algorithm decides the best parameter settings following the guidelines provided in section 3.5.

Figure 4 shows the algorithm. First, the algorithm is fired if the prediction value of the classifier oscillates, and the detected imbalance in class experiences exceeds a certain threshold ir_0 set by the user. This threshold indicates the maximum noise allowed in the classifier; its value may be set according to ϵ_0 . Then, the algorithm checks whether the classifier being updated is experienced enough and has higher numerosity than the population average. In this case, β and θ_{GA} are adapted.

Figure 5 shows the algorithm for the adaptation of β . Recall that β should be adjusted so that the real values of prediction and ϵ of that overgeneral classifier will be close to the theoretical ones (estimated with formula 6). Here the worst case is considered: we suppose that the classifier receives one example of the minority class, and then, $ir_{cl} = \frac{exp_{maj}}{exp_{min}}$ instances of the majority class. We compute $p_{ir_{\beta}}$ as the prediction value that the classifier would get after receiving these $ir_{cl} + 1$ examples with the current β value. So, $p_{ir_{\beta}}$ is calculated with the following series:

$$p_{ir_{\beta_0}} = R_{max} \cdot (1 - \beta) \quad (19)$$

$$\forall 1 < i \leq ir_{cl} : p_{ir_{\beta_i}} = \beta(R_{max} - p_{ir_{\beta_{i-1}}}) \quad (20)$$

If $p_{ir\beta}$ is still far from the theoretical estimate, we decrease β by a proportion $\zeta < 1$ and repeat the same process. In this way, β is adapted to guarantee that, in the worst case, the prediction and ϵ values will be close to the theoretical ones.

Finally, θ_{GA} is set according to equation 18. Specifically, f_{min} is estimated as:

$$\theta_{GA_t} = k_2 \cdot \frac{1}{f_{min}} = k_2 \cdot \frac{exp_{min} + exp_{maj}}{exp_{min}} \quad (21)$$

where k_2 determines the minimum number of examples of the minority class between GA events. For $k_2 = 1$ the system only sees one example of the minority class between GA events. In the experiments made in this paper, we used $k_2 = 5$ to allow for better stability of parameter estimates between GA events.

4.3 Results

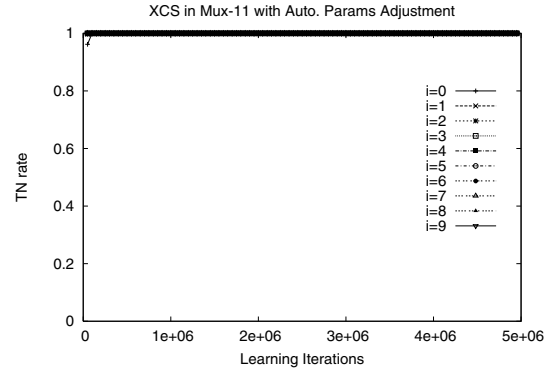
Figure 6 shows the results obtained in XCS with automatic adjustment of β and θ_{GA} . The results are similar to those shown in figure 3, where XCS can solve 11-bit multiplexer for imbalance levels up to $i=8$. With online parameter adaptation, the convergence is delayed because XCS needs some time to realize the existence of the overgeneral rules to delete them.

The parameter adaptation algorithm simplifies XCS's tuning, since it does not require a previous estimation of the learning rate and GA triggering threshold, which are critical to XCS's performance. There are few constants that must be set for the user, which may be initialized as follows (as runs shown herein):

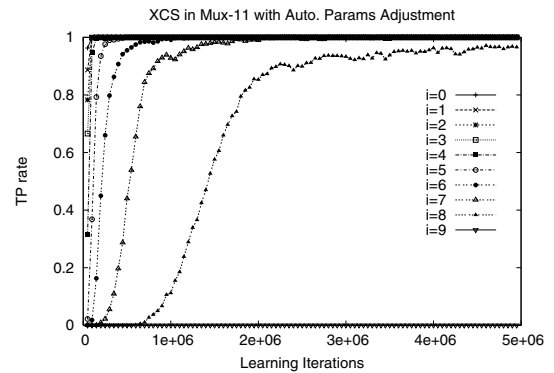
$$L_{max}=0.99, L_{dec}=0.9, ir_0=2000, \theta_{ir}=10000$$

L_{max} and L_{dec} serve to detect that the classifier's prediction oscillates. They could be replaced by checking if the classifier receives a counterexample. ir_0 is the maximum imbalance ratio permitted. We set $ir_0 = 2000$ according to the maximum bound obtained with formula 10. Finally, we set $\theta_{ir} = 5 \cdot ir_0$ to ensure that a classifier will receive 5 instances of minority class at the higher imbalance ratio before firing the algorithm.

Neither a manual nor an automatic adjustment permitted XCS to solve the multiplexer problem at imbalance levels $i=9$ and $i=10$. By setting appropriately β and θ_{GA} we assure that the parameters of overgeneral classifiers will be close to its theoretical values and the reproductive opportunities of all niches will be balanced. However, such a low supply of minority class instances obstructs the discovering of highly rewarded niches predicting the minority class. For example, for $i=10$ XCS sees about 1000 examples of the minority class in 1,000,000 learning iterations. Under a pure exploration regime, half of these 1000 examples activate niches of class 1 (highly rewarded niches), and the other half activate niches of class 0 (with reward 0). Thus, the system has only approximately 500 examples for discovering the high rewarding niches of the minority class. Such a severe imbalance ratio makes learning of the minority class really difficult. It is very hard to discover such niches in the population. Even though they are discovered, they are hard to maintain; with such low supply, the probability that a niche will be lost is very high. However, we believe that XCS is quite robust to high class imbalances. Imbalance levels $i = 9$ and $i = 10$ represent two extreme cases in which we suspect that many learners will also suffer from biases.



(a) TN rate



(b) TP rate

Figure 6: TN rate (a) and TP rate (b) in the 11-bit multiplexer with imbalance levels from 0 to 9. XCS was run with parameter adaptation on β and θ_{GA} .

5. DISCUSSION

The work presented here has to do with the discovery and maintenance of niches for unbalanced problems. We are really more concerned about the presence of *unbalanced niches* rather than the general approach to the class imbalance problem. A complexity factor for XCS is how the niches are distributed in the feature space. It is possible that a niche has a high number of instances compared to another one that is described by a few number of instances, even though the dataset has the same number of instances per class. If two niches with significantly different number of instances are close in the feature space, XCS could evolve a classifier overgeneralizing both niches simultaneously. This approach is in close relation with the small disjuncts problem described in [6]. The key issue is how to identify the presence of small disjuncts and avoid the tendency to overgeneralization. Our mechanism for parameter adaptation is a novel approach that takes advantage of XCS's learning mechanism; it does not need any a priori estimation of the small disjuncts problem because it uses the information collected by the classifiers themselves. We are currently investigating how the approach applies to other types of problems.

A systematic study of XCS's niche support for balanced

problems can be found in [3]. Therein, the authors suggest that the population size should be increased linearly with $1/p_{occ}$, where p_{occ} must be set to the lowest occurrence probability. Our experimental proofs demonstrated that this is not as important if β and θ_{GA} are tuned properly. Furthermore, not increasing the population size allows for less computational resources. We agree with the authors that a larger GA threshold balances the opportunities between frequent and infrequent niches. The proposed method for parameter adaptation uses a standard low GA threshold and high β value in the first iterations, which allows a fast estimation and speeds up learning, and later on adapts their values to stabilize learning.

Although a high value of θ_{GA} rebalances the reproduction opportunities of all niches, the low supply of examples of the minority class niches slows down the estimates of the corresponding classifiers. Therefore, it seems feasible to apply either an oversampling or an undersampling method to increase the occurrence of unfavored niches. Again, the key issue is how to detect those infrequent niches. The algorithm used to adapt XCS's parameters could be widened to oversample those examples triggering the least frequent niches.

The results show XCS's robustness with the unbalanced multiplexer problem. Our aim is to extend the study to a more general class of problems, containing continuous attributes, non-overlapping niches, and noise. Our estimation of the imbalance ratio in a niche assumes that there is no noise in the dataset. The presence of instances falling inside a niche which are wrongly labeled as the other class would be considered as unbalanced instances. In this case, XCS could overfit and evolve several niches instead of a single one. The setting of ϵ_0 and R_{max} determines the degree of noise admitted in a classifier; so that all noise falling below the threshold would be considered negligible.

6. CONCLUSIONS

We showed that XCS with standard parameter settings is quite robust to the unbalanced multiplexer problem up to an imbalance ratio of 64. For higher imbalance levels, an appropriate tuning of XCS's parameters is needed to achieve classification of infrequent examples.

We first studied theoretical imbalance ratio bounds that allow XCS to detect such imbalances, obtaining the threshold between what is to be considered as unbalanced proportion of examples and negligible noise. Online learning with a windowed update mechanism specially suffers from very infrequent examples. A high learning rate produces unstable parameter estimates and provides short memory to remember examples coming infrequently. Lower values of the learning rate parameter are needed although this consequently slows down convergence. In turn, slower convergence in parameter estimates suggests a decrease of GA frequency, provided that the GA is going to work with similar parameter estimates as in the balanced case. Moreover, occurrence-based reproduction is highly sensitive to imbalances: it tends to give higher reproduction opportunities to the most frequent niches, in a relation proportional to the imbalance ratio. Delaying the GA triggering mechanism is proved an effective method to counterbalance this effect.

We provided guidelines to set XCS's parameters based on the dataset imbalance ratio. In case that the imbalance ratio could not be estimated, we proposed a method to

adapt XCS's parameters based on the information collected by XCS during training. We tested the 11-bit multiplexer; first, by adjusting XCS's parameters following the guidelines, and further using the automatic adaptation method. Results showed that, with appropriate parameter settings, XCS is robust to class imbalances. As further work, we would like to test XCS on multiple step problems in which the problem of unbalanced niches is severe as some environmental states are rarely seen by the system.

For classification tasks, research is on the way to further improve XCS's performance and speed up convergence based on resampling techniques or additional mechanisms acting at the classifier level. At this stage of research, there are promising results to establish a comparison with other types of learners and extend the study to real-world problems.

Acknowledgements

The authors thank Stewart W. Wilson for the comments and feedback he provided throughout this research. We also thank the support of *Enginyeria i Arquitectura La Salle*, Ramon Llull University, as well as the support of *Ministerio de Ciencia y Tecnología* under project TIN2005-08386-C05-04, and *Generalitat de Catalunya* under Grants 2005FI-00252 and 2005SGR-00302.

7. REFERENCES

- [1] Bernadó-Mansilla, E. and Garrell, J.M. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evolutionary Computation*, 11(3):209–238, 2003.
- [2] Butz, M.V., Goldberg, D., and Tharankunnel, K. Analysis and improvement of fitness exploration in XCS: bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11(3):239–277, 2003.
- [3] Butz, M.V., Goldberg, D.E., Lanzi, P.L., and Sastry, K. Bounding the Population Size to Ensure Niche Support in XCS. Technical report, IlliGAI Report No. 2004033, July 2004.
- [4] Butz, M.V. and Wilson, S.W. An algorithmic description of XCS. In P. Lanzi, W. Stolzmann, and S. Wilson, editors, *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, volume 1996 of *Lecture Notes in Artificial Intelligence*, pages 253–272. Springer, 2001.
- [5] Holmes, J.H. Differential negative reinforcement improves classifier system learning rate in two-class problems with unequal base rates. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 635–642. Morgan Kaufmann, 1998.
- [6] Holte, R.C., Acker, L.E., and Porter, B.W. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 813–818. San Mateo, CA: Morgan Kaufmann, 1989.
- [7] Japkowicz, N. and Stephen, S. The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis*, 6(5):429–450, November 2002.
- [8] Orriols-Puig, A. and Bernadó-Mansilla, E. The Class Imbalance Problem in Learning Classifier Systems: A Preliminary Study. In F. R. et al., editor, *Genetic and Evolutionary Computation Conference (GECCO2005) workshop program*, pages 74–78, Washington, D.C., USA, 2005. ACM Press.
- [9] Orriols-Puig, A. and Bernadó-Mansilla, E. The Class Imbalance Problem in UCS Classifier System: Fitness Adaptation. In *Congress on Evolutionary Computation*, volume 1, pages 604–611, Edinburgh, UK, 2005. IEEE.
- [10] Wilson, S.W. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
- [11] Wilson, S.W. Generalization in the XCS Classifier System. In J. K. et al, editor, *Genetic Programming: Proceedings of the Third Annual Conference*, pages 665–674. Morgan Kaufmann, 1998.
- [12] Wilson, S.W. Personal Communication, July 2005.